

Description

[Reliably Continuing a Secure Connection When The Address of a Machine at One End of the Connection Changes]

BACKGROUND OF INVENTION

[0001] Field of the Invention

[0002] The present invention relates to telecommunication networks, and more specifically to a method and apparatus for reliably continuing a secure connection when the address of a machine at one end of the connection changes.

[0003] Related Art

[0004] Secure connections are often used to attain various features such as confidentiality/privacy (preventing third parties from accessing/viewing the exchanged information), data integrity (the data sent by a sender is not tampered with, by third parties), and authentication (the data is sent by the party purporting to be the sender), as is well

known in the relevant arts.

[0005] A security association (SA) is generally associated with each secure connection. A SA generally contains several parameters which define the security services offered to a secure connection. SA is described in further detail with reference to examples in RFC 2401, available from www.ietf.org. In the embodiments described below, SA is used to specify the authentication mechanism (e.g. usage of digital certificates from a specific certificate authority or passwords, cryptographic algorithm etc.) and the traffic encryption services (e.g. payload encapsulation or header authentication, etc.).

[0006] SA is generally bound to a machine address (e.g., IP address) of another end. Typically, protocols such as Internet Security Association and Key Management Protocol (ISAKMP, described in RFC 2408) provide for creation of an SA for a secure connection, which is then used for secure exchanges on the connection. Each end machine uses an IP address of the other end machine in associating the SA to the specific secure connection.

[0007] The address of a machine at one end of the secure connection may change. For example, the IP address of a mobile node (e.g., a laptop machine) may change when the

node moves from one sub_net (access network) to the other (in which common addresses would not be usable) as is well known in the relevant arts. It is often desirable that connectivity for user applications (operating on top of the secure connection) be continued seamlessly (with minimal disruption) when the address of a machine at one of the secure connection changes.

[0008] In one prior approach, the secure connection and the associated SA are set up afresh when the address of a machine at one end changes. Once the connection and the SA are set up, connectivity is continued for the user applications. In general, applications encounter disruption in connectivity while the secure connection and the SA are being set up.

[0009] Such a situation is undesirable generally when the users experience the disruption. The disruption may be experienced at least in situations when connectivity is absent for a few seconds. Such extended absence of connectivity can be encountered in situations where computationally intensive tasks may need to be repeated while negotiating various SA attributes.

[0010] For example, a substantial amount of processing resources may be required to generate keys corresponding

to a machine at one end of the secure connection, and accordingly the users may experience the disruption. The disruption may be particularly more with devices (such as notebooks, handheld personal digital assistant, etc.) having limited computational resources. In addition, it may be desirable to minimize/avoid the data transfer overhead resulting from the need to negotiate various SA attributes.

[0011] At least some of such requirements appear to be addressed by the teachings of a document entitled, "Address Management for IKE version 2", by Francis Dupont, Internet Engineering Task Force, publication date: January 2003 (hereafter "Dupont"). Specifically, in section "2.4 Mobility requirements" of Dupont, it is stated that, "When the mobile node moves, another care_of address is used and some SAs, including the IKE SA, must be updated to use the new address." Section "3.5 Explicit address update" of Dupont further states that "The explicit mechanism MUST be used. A new notification has to be defined. We propose to copy it from the delete notification." Delete notification is defined in RFC 2408 as well as in Annex C of a revised version of the above document with same title published in June 2003. The format specified there does not appear to include a provision for including the new

machine (IP) address of the end machine (moving to a new location). It appears that the other end machine would learn the new IP address from the header (source IP address) of the explicit address update packet.

[0012] Several problems may be presented by such an approach. For example, an unknown third machine may snoop the explicit address update packet and "replay" with a different source IP address. Replay generally entails using the payload of the explicit address update packet noted above and resending the same payload in another packet. The machine at the other end of the connection may accept the address update and start communicating with the unknown third machine (instead of the machine that has moved with a new IP address). That is, the packets intended for the moving end machine may thereafter be diverted to the unknown third machine, which is undesirable.

[0013] Dupont further teaches a 'reverse routability check' to address some of the issues associated with the replay_type possibility. 'Reverse routability check' generally refers to an approach in which the machine at the other end of the secure connection (i.e., the end machine other than the one with the changing IP address) may request the moving

end machine to identify itself, and the moving end machine may send a response with an appropriate identifier.

[0014] Unfortunately, reverse routability check type of approaches also may not be robust enough in avoiding the diversion of packets to unknown third machines. For example, when the unknown third machine (after sending the address update request) receives a reverse routability check, the contents of the check may be forwarded to the machine which has moved to a new address. The machine with the new address may respond back with an appropriate response for the reverse routability check, which can be forwarded to the machine at the other end of the secure connection. The response may be accepted by the machine at the other end, and packets may thereafter be diverted to the unknown third machine.

[0015] What is therefore needed is a method and apparatus to reliably continue a secure connection when the address of a machine at one end of the connection changes.

BRIEF DESCRIPTION OF DRAWINGS

[0016] The present invention will be described with reference to the accompanying drawings briefly described below.

[0017] Figure 1 is a block diagram illustrating an example environment in which the present invention can be imple-

mented.

[0018] Figure 2 is a flow chart illustrating the manner in which an end machine may allow connectivity to be continued on a secure connection even if the address of the end machine changes, according to an aspect of the present invention.

[0019] Figure 3 is a flow chart illustrating the details of a machine connected at an end of a secure connection which enables connectivity to be continued according to an aspect of the present invention when the address of a machine at the other end of the secure connection changes.

[0020] Figure 4 is a block diagram illustrating the details of an embodiment of an end machine which allows connectivity to be continued on a secure connection even if the address ('self_address') of the end machine changes.

[0021] Figure 5 is a block diagram illustrating the details of an embodiment of an end machine which enables connectivity to be continued according to an aspect of the present invention when the address of a machine at the other end of the secure connection changes.

[0022] Figure 6 is a block diagram illustrating the details of an embodiment of an end machine implemented substantially in the form of software according to an aspect of the present invention.

[0023] In the drawings, like reference numbers generally indicate identical, functionally similar, and/or structurally similar elements. The drawing in which an element first appears is indicated by the leftmost digit(s) in the corresponding reference number.

DETAILED DESCRIPTION

[0024] *1. Overview*

[0025] According to an aspect of the present invention, if the IP address of an end machine at one end of a secure connection changes, the new IP address of the end machine is included in the payload (non_header) of an IP packet sent to the other end machine. In one embodiment, the new IP address is included in the ISAKMP portion of the IP packet payload.

[0026] As the ISAKMP portion is encrypted and authenticated, undesirable third parties may not be able to decipher or alter the ISAKMP portion. Due to the inability to decipher the packet, 'replay' type options may not be possible. Accordingly, attacks such as those described above in the background section may be avoided.

[0027] The new IP address thus sent may be used associated with the same SA (security association) which was used for a

secure connection between two end machines prior to the change of IP address. As a result, the computational and/or data transfer requirements may be minimized in continuing connectivity for user applications after the change of address. Thus, connectivity can potentially be continued seamlessly with imperceptible disruption for users.

[0028] Several aspects of the invention are described below with reference to examples for illustration. It should be understood that numerous specific details, relationships, and methods are set forth to provide a full understanding of the invention. One skilled in the relevant art, however, will readily recognize that the invention can be practiced without one or more of the specific details, or with other methods, etc. In other instances, well known structures or operations are not shown in detail to avoid obscuring the invention.

[0029] *2. Example Environment*

[0030] Figure 1 is a block diagram of a networking system illustrating an example environment in which the present invention can be implemented. The networking system is shown containing client system 110, access networks 120_A and 120_B, Internet 150, gateway 160, service network 180, and server systems 190_A through 190_D. Each

system is described in further detail below.

[0031] Client system 110 represents an example end machine (of a secure connection), whose address can change. For illustration, the address is assumed to change as client system 110 is connected from different access networks (e.g., first via access network 120_A and then via access network 120_B). Accordingly, the same client system 110 is shown at two different places in Figure 1.

[0032] However, the address of end machines can change for other reasons as well. For example, when a mobile node moves from one access network to another, the IP address can also change as is well known in the relevant arts. In general, client systems are used to access various services (e.g., provided by server systems 190_A through 190_D) using secure connections managed according to various aspects of the present invention.

[0033] Server systems 190_A through 190_D provide various services (e.g., web pages, video conferencing facility, data access, email), and are connected to Internet 150 via service network 180. The services are accessed by various users using end systems as noted above.

[0034] Service network 180 provides the connectivity between the server systems and Internet 150. Service network 180

may correspond to a corporate network, and be implemented using protocols such as IP, ATM. In such a scenario, each user may be viewed as an employee accessing the corporate services (resources) over Internet 150.

[0035] Access networks 120_A and 120_B enable end systems to be connected to Internet 150 using technologies such as dial_up connections, DSL, etc. In general, each access network forces end systems to use a corresponding group of addresses (either sub_networks or networks), and thus different access networks are associated with different group of addresses. In general, the approaches described herein are applicable when an end machine moves from one sub_net (in which common addresses cannot be used) to the other.

[0036] Internet 150 provides connectivity between client systems 110/110_B and server systems 190_A through 190_D. Internet 150 contains several network devices (e.g., routers, switches, gateways, bridges, etc.,) which may be implemented to support protocols (e.g., TCP/IP) enabling data connectivity between various machines.

[0037] Gateway 160 operates cooperatively with client system 110_A (using protocols such as IPSec described in RFC 2401) to provide a secure connection to end users. In

general, the secure connection operates according to various attributes of a security association (SA), which are generally negotiated at the time the secure connection is set up. Gateway 160 and client system 110_A form the two end machines for such a secure connection. Gateway 160 may be contained within service network 180 such that a user is provided a secure connection terminating within the service network.

[0038] An aspect of the present invention enables secure connection to be continued in a seamless manner (i.e., without much disruption) reliably even if the address of client system 110_A changes when moving from access network 120_A to access network 120_B. Such a feature is implemented in the context of an environment operating generally according to RFCs 2401 (Security Architecture for the Internet Protocol), 2408 (Internet Security Association and Key Management Protocol (ISAKMP)), and 2409 (The Internet Key Exchange (IKE)).

[0039] The details of such an environment according to various aspects of the present invention are described below in further detail.

[0040] *3. Establishing Security Connection in General*

[0041] Broadly, in an embodiment, various attributes of a security

association are initially negotiated between two end machines before the data is transferred between the end systems and the server systems. For illustration, it is assumed that client system 110 ('initiator') initiates a first message (related to establishing a security connection) and gateway 160 ('responder') responds to the message sent by initiator. A security association is set_up by the end of the exchange phase. The exchange phase is described briefly below.

[0042] In one example mode (referred to as "main mode" in RFC 2409), the exchange phase contains a total of six messages grouped into three sub_phases: proposal negotiation, key exchange and authentication phase. Each of the sub_phase contains two messages, one message sent by initiator and another message sent by responder in response to message sent by the initiator. The contents of the messages are briefly described below.

[0043] In the proposal sub_phase, the initiator sends a message containing several suites of choices of authentication mechanisms (e.g., pre_shared secret, digital certificates, etc.), the parameters that will be used in the following key exchange messages (e.g., Oakley group numbers of Diffie_Hellman public key exchange), the cryptographic

hash that will be used to authenticate the messages (e.g., SHA_1 or MD5), the cryptographic algorithm that will be used to encrypt the exchange messages, lifetime of the secure connection, etc. Message 2 from responder contains one of the choice from the suites proposed in message 1, indicating the accepted authentication and encryption mechanism to be used during further communication.

[0044] In key exchange phase, initiator, in message 3, may send the Diffie_Hellman public value (as per the group number agreed on message 1 and 2) corresponding to the initiator. In response, responder sends message 4 which contains its own Diffie_Hellman public value. These public values may be used by both initiator and responder to arrive at a secret session key that is used to encrypt and decrypt the data exchanged between the initiator and responder from next message onwards.

[0045] In an authentication phase, both messages 5 and 6 may contain unique identifiers (called 'identity') and a proof of the identity. Such an identifier uniquely identifies the sender, and the proof of the identity can be a digital certificate or a cryptographic hash computed using the pre_shared secret as a parameter (as agreed on messages

1 and 2). Both messages 5 and 6 are encrypted using the secret session key that is derived from the D_H public values exchanged in messages 3 and 4.

[0046] Once the main mode exchange is over, the entities derive an ISAKMP SA, which contains information about authenticity of the peer (other end device), and a cryptographic context to communicate further with the peer. In general, setting up a ISAKMP SA entails substantial overhead, and it is desirable to avoid such set up at least for devices with limited processing power. An aspect of the present invention enables the ISAKMP SA to be used even after the self_address of an end device changes.

[0047] Multiple secure connections may then be established between the end devices using the ISAKMP SA thus set up. In general, the end devices may need to set up an IPSec SA via a 3 way message handshake known as 'quick mode' (described in RFC 2409 in further detail). The messages in quick mode exchange are described below in further detail.

[0048] In quick mode exchange, the initiator sends a message to the responder which contains a set of proposals describing the protocol of security (e.g AH or ESP), encryption algorithm (e.g. 3DES, AES), authentication mechanism (e.g.

MD5_HMAC or SHA_1_HMAC), and a set of traffic selectors (describing the flow/nature of traffic that is to be covered by the corresponding security protection).

[0049] The responder replies with the accepted proposal of the security services and traffic selector. The initiator completes the exchange by sending a third authenticated message, proving its liveness. After this quick mode exchange is complete, both the initiator and responder derive an SA known as IPSec SA, whose attributes are used to afford security services to the traffic specified in the IPSec SA.

[0050] In one embodiment, at least some of the attributes agreed upon in the ISAKMP and IPSec SAs form a corresponding SA at each of the end machines (i.e., client system 110 and gateway 160). Each end machine maintains a security association database (SAD), which binds the SA to the address of the other end machine. The SAD may store data representing several security associations (to corresponding end machines), and each SA_end machine pair may be uniquely identified by security parameter index (SPI). The SPI values in main mode messages are often referred to as cookies.

[0051] Once various attributes of a SA are negotiated, the at-

tributes are used to provide a secure connection. The description is continued with reference to a manner in which client system 110 may use same SA to continue to send data, when IP address of client system 110 changes.

[0052] 4. *Method in End Machine Whose Address Changes*

[0053] Figure 2 is a flow chart illustrating the manner in which an end machine may allow connectivity to be continued on a secure connection even if the address of the end machine changes, according to an aspect of the present invention. For illustration, the method is described with reference to Figure 1. However, the method may be implemented in other environments as well, without deviating from the scope and spirit of various aspects of the present invention. The method begins in step 210, in which control is immediately transferred to step 210.

[0054] In step 210, client system 110 (operating as one end machine) determines various attributes of a security association (SA) by communicating with the other end machine (i.e., gateway 160 in the present example). A set of messages may be exchanged according to ISAKMP as described above to determine the SA attributes.

[0055] In step 220, client system 110 sends packets on a secure connection using the SA attributes determined above. The

packets may be sent on the secure connection using the determined SA attributes in a known way.

[0056] In step 230, client system 110 detects a change of self_address (from an old address to a new address). In Figure 1, the change occurs when client system 110 moves from access network 120_A to access network 120_B. A new IP address may be assigned to client system 110 by access network 120_B, and the change in IP address may be detected by client system 110.

[0057] In step 240, client system 110 sends a request (address update packet) to gateway 160 requesting the new address to be bound to the SA already present. The new address is contained in a payload portion (e.g., as a part of ISAKMP portion) of an IP packet, as described in a section below. By including the packet in the payload additional security mechanisms such as encryption and authentication can be used to ensure that the new IP address and the payload cannot be used by unknown third parties.

[0058] In step 260, client system 110 determines whether an acceptance is received from gateway 160 indicating that the new address is bound to the SA. In an embodiment, if a response is not received within a pre_specified time, an acceptance is deemed to not have been received. An ex-

PLICIT reject message may be received as well within the pre_specified time.

[0059] Control is transferred to step 290 if the new address is bound, otherwise to step 280. In step 280, client system 110 re_negotiates SA attributes as a change of address (of client system 110) for the secure connection is not supported by gateway 160. Such re_negotiation may be performed in a known way. Control is then transferred to step 299.

[0060] In step 290, client system 110 continues communication with the same security associations set up in step 220. Thus, packets are sent using the new address of client system 110, but with the same SA attributes determined in step 220.

[0061] As a result, the overhead of re_negotiating (including any computations required for generating keys and transmission overhead) new SA attributes is avoided. Control is then transferred to step 299, in which the method ends.

[0062] The description is continued with reference to a method by which gateway 160 enables the same SA to be used when the address of client system 110 changes according to an aspect of the present invention.

[0063] *5. Method in Gateway*

[0064] Figure 3 is a flow chart illustrating the manner in which a machine at an end of the secure connection enables the same SA to be used when the address of a machine at the other end of the secure connection changes according to an aspect of the present invention. For illustration, the method is described with reference to Figure 1. However, the method may be implemented in other environments as well. Such implementations are also contemplated to be within the scope and spirit of various aspects of the present invention. The method begins in step 301, in which control immediately passes to step 310.

[0065] In step 310, gateway 160 (operating as one end machine) determines various attributes of a security association (SA) by communicating with the other end machine (i.e., client system 110 in the present example). As noted above, set of messages may be exchanged according to ISAKMP to determine the SA attributes.

[0066] In step 320, gateway 160 binds the SA to the address of client system 110 at the other end of the secure connection. In step 330, gateway 160 receives a request indicating a change of address of client system 110 at the other end of the secure connection. In the illustrative example, the changed (or new) address corresponds to an IP ad-

dress assigned by access network 120_B. The IP address is contained in a payload portion of a packet forming the request.

[0067] In step 340, a determination is made as to whether the binding of security association can be changed to the new address. In general, such a determination may be performed based on absence of support for change of binding or based on configuration data supplied by a user. Control is transferred to step 350 if such a change of binding is possible, otherwise to step 380.

[0068] In step 350, gateway 160 binds the security association (SA) to changed IP address of client system 110. In an embodiment, the security association database (SAD) is updated to reflect that the existing SA is to be associated with the new/changed IP address.

[0069] In step 355, gateway 160 determines whether the binding is successful. Control passes to step 360 if the binding is successful, or else control passes to step 390. In step 390, a reject response (indicating that binding is not performed) may be sent to client system 110. Control then passes to step 399.

[0070] In step 360, gateway 160 sends a response indicating that the new address is now bound to the pre_existing SA. In

step 370, gateway 160 continues communication with the same security associations (as stored in SAD), but using the new address of client system 110. Thus, the network layer connectivity continues by using the new address, and using the same SA reduces the computational overhead and disruption to connectivity. Control is then transferred to step 399.

[0071] In step 380, gateway 160 merely ignores the request received in step 330. In such a case, end_system 110 re_negotiates SA attributes, as the same SA may not be used for data transfer using the new address of client system 110. Control is then transferred to step 399 in which the method ends.

[0072] From the above, it may be appreciated that the two end machines need to exchange messages to enable the same SA to be used with the new address of one of the end machines. The description is continued with reference to example packet formats which facilitate such an exchange.

[0073] *6. Packet Formats*

[0074] In an embodiment, client system 110 and gateway 160 exchange packets consistent with (or extending) ISAKMP protocol described in further detail in RFC 2408. As noted in RFC 2408, the 'Notification Payload' contains a 'Notify

Message Type' (Bytes 11 and 12), and values (8192 _ 16383 and 32768 _ 40959) are reserved for private use.

[0075] Three new types of messages (IPADDRESS_UPDATE, IPADDRESS_UPDATE_ACK and IPADDRESS_UPDATE_NACK) are defined using the corresponding three values reserved for private use. An IPADDRESS_UPDATE packet is used by client system 110 to send a request indicating the new address. IPADDRESS_UPDATE_ACK and IPADDRESS_UPDATE_NACK packet types are used by gateway 160 to indicate acceptance and rejection of the request respectively.

[0076] If IPADDRESS_UPDATE packet type is not recognised by gateway 160, the packet may simply be ignored as noted in step 380 above. IPADDRESS_UPDATE may be designed with the following format: Bytes 13 to 16: The old/previous IP address; Bytes 17 to 20: The new/changed IP address; Bytes 21 to 24: Unique identifier of the request; Byte 21: Number of SPIs; and Bytes 22 onwards: List of SPIs.

[0077] It may be appreciated that a single IPADDRESS_UPDATE packet may contain a list of SPIs, with each SPI uniquely identifying a SA at the other end (i.e., in gateway 160). Thus, when multiple secure connections are present to the

same end device (but possibly connecting to different server systems), a single update packet may request the address change with respect to all the corresponding security associations.

[0078] IPADDRESS_UPDATE_ACK packet type may repeat the content noted above (bytes 13 onwards) if the new address is bound to all the security associations specified by the list of SPIs. If the request is accepted with respect to only some of the SPIs, only those SPIs may be included in the ACK packet type. Similarly, the IPADDRESS_UPDATE_NACK packet type may repeat the contents noted above if the new address cannot be bound to any of the security associations specified by the list of SPIs.

[0079] Thus, the packet formats described above can be used to implement a change of IP address bound to a security association (SA). It should be understood that more elaborate approaches requiring multiple communications before changing an address binding to a SA may be implemented, as will be apparent to one skilled in the relevant arts by reading the disclosure provided herein. Similarly, while negotiating SA attributes, the end machines may exchange packets to determine whether the other end supports binding of an SA to a new address. In such a case,

step 380 may be avoided.

[0080] From the above, it may be appreciated that the new IP address is contained in the ISAKMP portion of the IP payload. As is well known, encryption and authentication is generally applied on the payloads of the ISAKMP packets. This encryption and authentication mechanisms are generally negotiated during ISAKMP negotiation. For example, Advanced Encryption Standard (AES) cipher can be used to encrypt the payloads and Secure Hash Standard (SHA_1) can be used to authenticate them. By using such encryption and authentication techniques third party attacks can be avoided.

[0081] In general, ISAKMP portion contains several sub_portions referred to as ISAKMP payloads, with respective ISAKMP payloads being used to include authentication and encryption related data. Encryption ensures that confidentiality is preserved, and authentication ensures that the source of the packet is reliably communicated to the recipient.

[0082] Thus, by including the new address (and other notification information) in the payload of a packet, the present invention prevents unknown third systems from diverting traffic destined to an end system which moves to a new

address. The description is now continued with respect to example embodiments of client systems and gateways.

[0083] *7. Client System*

[0084] Figure 4 is a block diagram illustrating the details of an embodiment of client system 110 as relevant to various aspects of the present invention. Client system 110 is shown containing inbound interface 410, parser block 430, IP address block 440, application block 450, ISAKMP block 460, SA tables 465, IPSec block 470, and outbound interface 490. Each component is described in further detail below.

[0085] Inbound interface 410 provides the electrical, physical, and protocol interfaces to receive data packets from various systems and end devices (including gateway 160). The received packets are forwarded to parser 430 in appropriate format (e.g., IP packets). Similarly, outbound interface 490 provides the electrical, physical, and protocol interfaces to send IP packets to various systems and end devices. Both inbound interface 410 and outbound interface 490 may be implemented in a known way.

[0086] Parser 430 examines each received packet and forwards the received packet to one of ISAKMP block 460 and IPSec block 470. The specific block to forward to depends gen-

erally on the header contents (e.g., protocol field, destination port number, etc.) of each received packet. Parser 430 may be implemented in a known way.

[0087] Application block 480 may generate data packets related to user applications (executed on client system 110). The data packets are forwarded to IPsec block 470 for transmission on a secure connection. Similarly, data packets, which are received on a secure connection, are received from IPsec block 470. Application block 480 provides various user applications such as file sharing, email, voice over IP, video conferencing, etc., and may be implemented in a known way. As may be appreciated, the data related to such applications continue to be sent even after the self_address of client system changes.

[0088] IPsec block 470 uses the information available in SA tables 465 to provide a secure connection for application block 450. Thus, depending on the specific flow (source/destination address and source/destination port combination), the attributes of the corresponding SA are used to process the packet. Such processing may entail, among other tasks, encryption (according to a SA attribute) and transmitting the encrypted packet (using another SA attribute). Similarly, the packets received on a

secure connection may also be decrypted and forwarded to application block 450. IPsec block 470 may be implemented in a known way.

[0089] IP address block 440 determines a situation in which the self_address (i.e., the address on the interface 410) has changed, and interfaces with ISAKMP block 460 to cause the other end machine(s) to change the IP address binding (from old address to new address) to the security associations. IP address block 440 may pass to ISAKMP block 460 data representing the new address and the specific interface for which the address has changed.

[0090] ISAKMP block 460 sets up a secure connection by negotiating various SA attributes with an end device at the other end of the connection. The negotiation can be implemented in a known way. The SA attributes and the associated address of the end device at the other end are stored in SA tables 465 (which may store the entire data related to the security association database (SAD) noted above). In general, ISAKMP block 460 represents a block which sets up and manages a secure connection and is generally referred to as a connection management block.

[0091] In addition, ISAKMP block 460 interfaces with the other end device to cause re_binding of a new address to an ex-

isting SA. The packet formats described above may be used during such interfacing. Thus, when an indication of a old/new address combination is received from IP address block 440, ISAKMP block 460 examines SA tables 465 to determine all the SAs (SPI list) bound to the specific old address. ISAKMP block 460 may then construct an IPADDRESS_UPDATE containing the SPI list, and send the packet using outbound interface 490.

[0092] When a packet of IPADDRESS_UPDATE_ACK type is received from parser, ISAKMP block 460 updates SA tables 465 to reflect the changed bindings. On the other hand, if an acceptance is not received (including reception of a reject response), ISAKMP block 460 may remove the SA information from SA tables 465, and negotiate the SA attributes again. The implementation of ISAKMP block 460 will be apparent to one skilled in the relevant arts by reading the disclosure provided herein.

[0093] Thus, the embodiment of Figure 4 operates to use the same SA even after the self_address (of client system 110) changes. However, a cooperating implementation is required at the other end device also. The implementation of such an end device is described below with reference to Figure 5.

[0094] 8. *Gateway*

[0095] Figure 5 is a block diagram illustrating the details of an embodiment of gateway 160 as relevant to various aspects of the present invention. Gateway 160 is shown containing inbound interface 510, parser 520, routing block 530, forwarding block 540, ISAKMP block 560, SA tables 565, IPsec block 570, IP address block 580, and outbound interface 590. Each component is described in further detail below.

[0096] Inbound interface 510 and outbound interface 590 may be implemented similar to inbound interface 410 and outbound interface 590 respectively. Parser 520 examines each received packet and forwards the received packet to one of routing block 530, ISAKMP block 560 and IPsec block 570. The specific block to forward to depends generally on the header contents, as noted above with respect to parser 430. Parser 520 also may be implemented in a known way.

[0097] Routing block 530 receives packets related to various routing protocols (OSPF, RIP, etc.) and populates the entries in forwarding tables 535. The entries in forwarding tables 535 may be configured manually as well. In general, the entries specify the manner (next hop address

and/or specific interface provided on outbound interface 590) in which each IP packet is to be forwarded/processed. Routing block 530 may be implemented in a known way.

[0098] Forwarding block 540 receives packets from parser 520 and IPsec block 570, and forwards each received packet based on the destination address contained in the packet. In general, the specific interface on outbound interface 590 (or via IPsec block 570) on which to transmit a packet, is determined based on the routing entries in forwarding tables 535. When packets (e.g., received from server system 190_A) are to be transmitted on a secure connection, each packet is forwarded to IPsec block 570.

[0099] IPsec block 470 uses the information available in SA tables 565 to provide security services for data packets received from various end systems (e.g., server systems 190_A through 190_D) connected to gateway 160. Thus, depending on the specific flow (source/destination address and source/destination port combination) on which a data packet is to be sent, the attributes of the corresponding SA are used to encrypt and transmit a packet using outbound interface 590.

[0100] Similarly, the packets received on a secure connection

may also be decrypted and forwarded to various end systems (e.g., server systems 190_A through 190_D) via forwarding block 540. Packets may be transmitted within any security frame work such as IPsec, and accordingly IPsec block 470 represents an example implementation of a secure transmission block. IPsec block 570 may be implemented in a known way.

[0101] ISAKMP block 560 sets up a secure connection by negotiating various SA attributes with an end device at the other end of the connection. The negotiation can be implemented in a known way. The SA attributes and the associated address of the end device at the other end are stored in SA tables 565 (which may store the entire data related to the security association database (SAD) noted above).

[0102] ISAKMP block 560 receives an IPADDRESS_UPDATE packet (containing a new address in the ISAKMP portion of an IP payload) and determines whether change in the binding of a prior SA to the new address is possible. The entries in SA tables 565 are updated to reflect the binding of the new address to the pre_existing SA. ISAKMP block 560 then generates an IPADDRESS_UPDATE_ACK packet indicating acceptance of the request.

[0103] ISAKMP block 560 generates an IPADDRESS_UPDATE_NACK

packet if the binding cannot be changed for whatever reason. The implementation of ISAKMP block 560 will be apparent to one skilled in the relevant arts by reading the disclosure provided herein.

[0104] Thus, the embodiment of Figure 5 can be used to enable an end machine at the other end of the secure connection to use the same SAs to continue to send data (potentially without disruption) when the address (of end machine at the other end) changes.

[0105] It should be understood that the different components of client system 110 and gateway 160 can be implemented in a combination of one or more of hardware, software and firmware. In general, when throughput performance is of primary consideration, the implementation is performed more in hardware (e.g., in the form of an application specific integrated circuit).

[0106] When cost is of primary consideration, the implementation is performed more in software (e.g., using a processor executing instructions provided in software/firmware). Cost and performance can be balanced by implementing the devices with a desired mix of hardware, software and/or firmware. Embodiments implemented substantially in software is described below.

[0107] *9. Software Implementation*

[0108] Figure 6 is a block diagram illustrating the details of system 600 representing client system 110 or gateway 160 in one embodiment. System 600 is shown containing processing unit 610, random access memory (RAM) 620, secondary memory 630, output interface 660, packet memory 670, network interface 680 and input interface 690. Each component is described in further detail below.

[0109] Input interface 690 (e.g., interface with a key_board and/or mouse, not shown) enables a user/administrator to provide any necessary inputs to system 600. Output interface 660 provides output signals (e.g., display signals to a display unit, not shown), and the two interfaces together can form the basis for a suitable user interface for an user/administrator to interact with system 600.

[0110] Network interface 680 may enable System 600 to send/receive data packets to/from various other systems on corresponding paths using protocols such as Internet protocol (IP). Network interface 680, output interface 660 and input interface 690 can be implemented in a known way.

[0111] RAM 620, secondary memory 630, and packet memory 670 may together be referred to as a memory. RAM 620 receives instructions and data on path 650 (which may

represent several buses) from secondary memory 630, and provides the instructions to processing unit 610 for execution. In addition, the SAD (SA database) described above may be maintained in the memory.

[0112] Packet memory 670 stores (queues) packets waiting to be forwarded (or otherwise processed) on different ports. Secondary memory 630 may contain units such as hard drive 635 and removable storage drive 637. Secondary memory 630 may store the software instructions and data, which enable System 600 to provide several features in accordance with the present invention.

[0113] Some or all of the data and instructions may be provided on removable storage unit 640 (or from a network using protocols such as Internet Protocol), and the data and instructions may be read and provided by removable storage drive 637 to processing unit 610. Floppy drive, magnetic tape drive, CD-ROM drive, DVD Drive, Flash memory, removable memory chip (PCMCIA Card, EPROM) are examples of such removable storage drive 637.

[0114] Processing unit 610 may contain one or more processors. Some of the processors can be general purpose processors which execute instructions provided from RAM 620. Some can be special purpose processors adapted for spe-

cific tasks (e.g., for memory/queue management). The special purpose processors may also be provided instructions from RAM 620.

[0115] In general, processing unit 610 reads sequences of instructions from various types of memory medium (including RAM 620, storage 630 and removable storage unit 640), and executes the instructions to provide various features of the present invention described above.

[0116] It may be appreciated that the features of above may be useful in several situations. For example, a seamless connectivity may be provided between end machines connected by a secure connection when the address of an end machine at one end of the secure connection changes (while moving from one access_net to another).

[0117] Even though the description of above is provided with reference to a situation in which the two end devices (at either of the secure connection) contain an end system and a gateway, various aspects of the present invention are intended to cover other combinations (such as two gateways, two end machines, etc.). Similarly, the approaches can be extended to multicast scenario as well.

[0118] *10. Conclusion*

[0119] While various embodiments of the present invention have

been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present invention should not be limited by any of the above_described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.